Parallel Cyclic Wavefront Algorithms for Solving Semidefinite Lyapunov Equations *

José M. Claver[1], Vicente Hernández[2] and Enrique S. Quintana-Ortí[1]

[1] Depto. de Informática, Universitat Jaume I, E-12080 Castellón, Spain.
[2] Depto. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, E-46071 Valencia, Spain.

Abstract. In this paper, we describe new parallel cyclic wavefront algorithms for solving the semidefinite discrete-time Lyapunov equation for the Cholesky factor using Hammarling's method by the message passing paradigm. These algorithms are based on previous cyclic and modified cyclic algorithms designed for the parallel solution of triangular linear systems. The experimental results obtained on an SGI Power Challenge show a high performance for large scale problems and better scalability than previous wavefront algorithms for solving these equations.

1 Introduction

Discrete-time Lyapunov equations arise in a great variety of problems of control theory and signal processing like, e. g., model reduction of linear control systems by means of the design of balanced realizations, Hankel-norm approximation problems, frequency domain approximation problems, solution of Riccati equations using Newton's method, etc. [12, 14].

Among the different solvers for these equations [2, 6], Hammarling's algorithm [7] is specially appropriate for model reduction via balanced realizations, as it directly computes the Cholesky factor of the solution. All these methods present a cubic computational cost and, already for medium-size problems, require the use of parallel computers. Thus, several wavefront algorithms based on Hammarling's algorithm have been implemented for shared memory multiprocessors in [3, 10].

When dealing with large-scale problems, parallel distributed memory multiprocessors present the advantage of their scalability. In the last few years, parallel algorithms have been proposed for solving triangular linear systems on this type of architectures [8, 5, 13]. These parallel algorithms can be classified as fan-in/fan-out, wavefront, and cyclic algorithms. Following these ideas, parallel distributed fan-in/fan-out and cyclic algorithms, based on the Schur method [2] or the Hessenberg-Schur method [6], have been developed for solving Sylvester equations [11]. More recently, parallel distributed wavefront Lyapunov solvers, based on Hammarling's method, have been presented in [4].

The cyclic algorithms potentially offer the best performance, due to their minimal communication, on parallel distributed memory multiprocessors with a high latency. However, their performance deteriorates when the number of processors is increased. In this paper we describe several modifications of the cyclic algorithms based on the combination of cyclic and wavefront algorithms to overcome this deficiency.

In section 2 we review Hammarling's algorithm and its data dependency graph. In section 3 we present the parallel cyclic algorithms and, in section 4, a new sort of parallel cyclic wavefront algorithms. In section 5 we report experimental results on message passing based multiprocessors. Finally, the conclusions of the work are outlined in section 6.

* This research was partially supported by the CICYT Project TIC96-1062-C03-01-03.

2 Hammarling's Method

Consider the semidefinite discrete-time Lyapunov equation,

$$\bar{A}\bar{X}\bar{A}^T - \bar{X} + \bar{B}\bar{B}^T = 0, \tag{1}$$

where $\bar{A} \in \mathbb{R}^{n \times n}$ is the coefficient matrix, $\bar{B} \in \mathbb{R}^{n \times m}$ is part of the right-hand side matrix, $\bar{B}\bar{B}^T$, and $\bar{X} \in \mathbb{R}^{n \times n}$ is the matrix of unknowns. Hereafter, we assume that $n \leq m$. Note that in case $n > m$, it is possible to apply the same algorithm described in [7]. If the eigenvalues of matrix $\bar{A}$, denoted by $\{\lambda_1, ..., \lambda_n\}$, satisfy $|\lambda_i| < 1, i = 1, 2, \ldots, n$, then a unique, non-negative definite solution matrix $\bar{X}$ exists. In such case, it is possible to obtain the Cholesky decomposition of the solution $\bar{X} = \bar{L}\bar{L}^T$, where $\bar{L} \in \mathbb{R}^{n \times n}$ is a lower triangular matrix. However, using Hammarling's algorithm [7], equation (1) can also be solved directly for the Cholesky factor $\bar{L}$.

In the first step of Hammarling's algorithm, equation (1) is transformed into a (simpler) *reduced Lyapunov equation.* For this purpose, the real Schur decomposition of $\bar{A}$ is computed as $\bar{A} = QSQ^T$. Here, $Q \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $S \in \mathbb{R}^{n \times n}$ is a block lower triangular matrix with 1×1 and 2×2 diagonal blocks. Each 1×1 block contains a real eigenvalue of the coefficient matrix $\bar{A}$, and each 2×2 block is associated with a pair of complex conjugate eigenvalues. Algorithms for computing the real Schur decomposition on parallel computers are described in [1, 9].

Applying the orthogonal similarity transformations defined by Q, we obtain the reduced Lyapunov equation

$$SXS^T - X = -BB^T,$$

where $X = Q^T\bar{X}Q$ and $B = Q^T\bar{B}$. Next, the product BB^T is reduced to a simpler form by computing an LQ factorization of B,

$$B = \begin{pmatrix} G & 0 \end{pmatrix} P,$$

where $G \in \mathbb{R}^{n \times n}$ is lower triangular and $P \in \mathbb{R}^{m \times m}$ is orthogonal. Finally, the solution L of the reduced Lyapunov equation

$$S\left(LL^T\right)S^T - \left(LL^T\right) = -GG^T, \tag{2}$$

provides the Cholesky factor of the original equation as $\bar{L} = QL$.

2.1 *The Serial Algorithm*

Following the method described in [7], the matrices S, L and G in (2) are initially partitioned as

$$S = \begin{pmatrix} s_{11} & 0 \\ \mathbf{s} & S_1 \end{pmatrix}, \quad L = \begin{pmatrix} l_{11} & 0 \\ \mathbf{l} & L_1 \end{pmatrix}, \quad \text{and} \quad G = \begin{pmatrix} g_{11} & 0 \\ \mathbf{g} & G_1 \end{pmatrix}, \tag{3}$$

where s_{11} is either a scalar or a 2×2 block. In the scalar case, l_{11} and g_{11} are also scalars, and $\mathbf{s}$, $\mathbf{l}$, and $\mathbf{g}$ are column vectors of $n-1$ elements. Otherwise, s_{11}, l_{11}, and g_{11} are 2×2 blocks, and $\mathbf{s}$, $\mathbf{l}$, and $\mathbf{g}$ are $(n-2) \times 2$ blocks.

For the sake of simplicity, hereafter we assume that all the eigenvalues of S are real. This problem will be denoted as the *real case* of the Lyapunov equation. Hence, the next three equations are obtained from (2) and (3)

$$\begin{aligned} l_{11} &= g_{11}/\sqrt{1 - s_{11}^2}, \\ (s_{11}S_1 - I_{n-1})\,\mathbf{l} &= -\alpha\mathbf{g} - \beta\mathbf{s}, \qquad \text{and} \\ S_1\left(L_1L_1^T\right)S_1^T - \left(L_1L_1^T\right) &= -\tilde{G}\tilde{G}^T = -G_1G_1^T - \mathbf{y}\mathbf{y}^T, \end{aligned} \tag{4}$$

where $\alpha = g_{11}/l_{11}$, $\beta = s_{11}l_{11}$, $\mathbf{y} = \alpha\mathbf{v} - s_{11}\mathbf{g}, \mathbf{v} = S_1\mathbf{l} + sl_{11}$, and I_{n-1} stands for the identity matrix of order $n-1$.

The diagonal element l_{11} is directly computed from the first equation in (4). The lower triangular linear system in the second equation is then solved for $\mathbf{l}$ by forward substitution. Finally, the last equation is a discrete-time Lyapunov equation of order $n-1$, where $\tilde{G}$ has the following structure

$$\tilde{G} = \left(G_1, \mathbf{y} \right);$$

that is, $\tilde{G}$ is a block matrix composed of an $(n-1) \times (n-1)$ lower triangular matrix, G_1, and an $n-1$ column vector $\mathbf{y}$. Therefore, it is possible to obtain the Cholesky decomposition of the product $\tilde{G}\tilde{G}^T$ using the LQ factorization,

$$\tilde{G} = \left(\bar{G}\ 0 \right) \bar{P},$$

where $\bar{G} \in \mathrm{IR}^{(n-1)\times(n-1)}$ is lower triangular and $\bar{P} \in \mathrm{IR}^{n\times n}$ is orthogonal. This procedure can be repeated with the reduced Lyapunov equation,

$$S_1 \left(L_1 L_1^T \right) S_1^T - \left(L_1 L_1^T \right) = -\bar{G}\bar{G}^T,$$

of order $n-1$, until the problem is completely solved. Fig. 1 presents an algorithmic description of Hammarling's method.

```
Algorithm SH.
  for $j = 1 : n$
    1. Compute the diagonal element.
       $\alpha := \sqrt{1 - S(j,j)^2}; L(j,j) := G(j,j)/\alpha; \beta := L(j,j) \cdot S(j,j)$
    2. Solve the lower triangular linear system for $\mathbf{l}$.
       for $i = j+1 : n$
         $L(i,j) := \frac{\left(\alpha G(i,j) + \beta S(i,j) + S(j,j) \sum_{k=j+1}^{i-1} S(i,k)L(k,j)\right)}{(1 - S(i,i)S(j,j))}$
       end for
    3. Compute the vector $\mathbf{y}$.
       for $i = j+1 : n$
         $\mathbf{y}(i) := \alpha \left( L(j,j)S(i,j) + \sum_{k=j+1}^{i} S(i,k)L(k,j) \right) - S(j,j)G(i,j)$
       end for
    4. Compute the Cholesky factor of the matrix $G$ of order $n - j$.
       for $i = j+1 : n$
         4.1. Compute the Givens rotation $(\sin\theta_i, \cos\theta_i)$ such that:
              $\left[ G(i,i)\ \mathbf{y}(i) \right] \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix} = \left[ *\ 0 \right]$
         4.2 Apply the Givens rotation.
              for $k = i : n$
                $\left[ G(k,i)\ \mathbf{y}(k) \right] := \left[ G(k,i)\ \mathbf{y}(k) \right] \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix}$
              end for
       end for
  end for
end SH
```

Fig. 1. Hammarling's serial algorithm.

2.2 *Study of the Data Dependencies.*

Hammarling's algorithm is column-oriented; that is, when the j-th column of the solution is to be computed, it is necessary to obtain the elements $L(j : i-1, j), j \leq i$, before computing the element $L(i, j)$. Consider now the computation of the $(j+1)$-th column of L. The first element that must be computed is $L(j+1, j+1)$ but, according to step 1 of the serial algorithm, $G(j+1, j+1)$ is required in iteration j to nullify the $(j+1)$-th element of $\mathbf{y}$. The next element to be computed is $L(j+2, j+1)$, which requires $L(j+1, j+1)$ and the updated element $G(j+2, j+1)$. Following this process, the data dependencies for solving a 4×4 discrete-time Lyapunov equation is shown in Fig. 2.

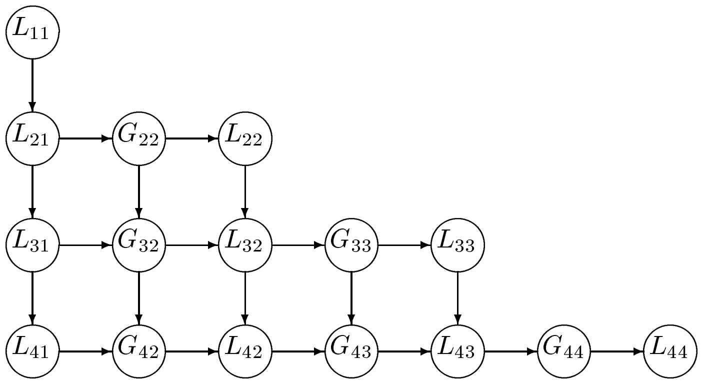

Fig. 2. Data dependency graph for a 4×4 Lyapunov equation.

It is important to outline from the analysis of the data dependencies, that the highest inherent parallelism is achieved when the elements on the same antidiagonal of L are computed simultaneously. This idea was previously introduced by O'Leary [15] in the context of the Cholesky decomposition problem and it was also used to design triangular linear system solvers on distributed memory multiprocessors in [8]. In [3, 10], this strategy was used to solve Lyapunov equations by Hammarling's method on shared memory multiprocessors.

3 Parallel Cyclic Algorithms

The parallel cyclic algorithms described in this paper are based on previous work by Heath and Romine [8], and Eisenstat *et al.* [5] for solving triangular linear systems on distributed memory multiprocessors. In these algorithms the matrices of the problem are partitioned and distributed among the processors by rows (or columns). This data layout presents good load balancing properties for matrix factorization procedures that generally precede the solution of triangular linear systems. In order to simplify the presentation of our algorithms we define a function $\mathbf{map}(j)$ that indicates the processor which stores the j-th row (column) of a matrix.

In the cyclic parallel triangular linear system solvers, all necessary information is stored in a segment of constant size $p-1$ that circulates among a ring of p processors. After receiving the segment, processor j computes an element of the solution. Next, the segment is updated and sent to the next processor $(j+1)$. The goal here is to overlap the circulation of the segment with the updating of several variables needed to compute the next element belonging to the same processor. Other kind of interconnection topologies like a bus can be appropriate for these algorithms since, as only one message is circulating at each time, no communication bottleneck exists.

Using the same ideas, Lyapunov equations can be solved by columns, with all the processors collaborating to compute each column. The solution of a column is divided into two stages: In the first stage, denoted as *substitution*, all the elements of a column are computed by solving a triangular linear system. In the second stage, denoted as *triangularization,* the right-hand side matrix $\tilde{G}$ is updated by computing an LQ factorization that nullifies vector $\mathbf{y}$. In this algorithm, the solution matrix L is stored cyclically by rows, like matrices S and G.

The proposed algorithm, *PHCF*, is shown in Fig. 3. In the *substitution* stage, the segment consists of the last $p-1$ computed elements of a column of L, and during segment circulation the unknown elements not computed are updated. In the *triangularization* stage the segment stores $2(p-1)$ elements with the $p-1$ sines and $p-1$ cosines computed. The circulation of the segment must be overlapped with the updating of the right-hand side matrix G and the vector $\mathbf{y}$.

```
Algorithm PHCF.
Memory: S(n/p,n), G(n/p,n), L(n/p,n), seg(n), Cos(n), Sin(n)
for $j = 0 : n-1$
1. Substitution stage of $L(:,j)$
   1.1. Compute the diagonal element.
        broadcast($L(j,j), S(j,j)$)
   1.2. Compute the subdiagonal elements.
        for $i = j+1 : n-1$
        1.2.1. Receive the segment $seg(p-1)$.
               receive(map($i-1$), $seg(\max(i-p+1, j+1) : i-1)$)
        1.2.2. Compute $L(i,j)$.
        1.2.3. Send updated segment.
               send(map($i+1$), $seg(\max(i-p+2, j+2) : i)$)
        1.2.4. Update $L(:,j)$.
        end for
2. Triangularization stage.
   2.1. Compute vector $\mathbf{y}$.
   2.2. Update matrix $G$.
        for $i = j+1 : n-1$
        2.2.1. Receive segments Sin and Cos.
               receive(map($i-1$), $Cos(\min(i-p, j+1) : i-1)$)
               receive(map($i-1$), $Sin(\min(i-p, j+1) : i-1)$)
        2.2.2 Apply the previous Givens rotations.
        2.2.3. Compute the Givens rotations $(\sin\theta_i, \cos\theta_i)$
        2.2.4. Send segments Sin and Cos.
               send(map($i+1$), $Cos(\min(i-p+1, j+2) : i-1)$)
               send(map($i+1$), $Sin(\min(i-p+1, j+2) : i-1)$)
        2.2.5. Apply the Givens rotations to the rest of elements.
        end for
end for
end PHCF.
```

Fig. 3 Algorithm *PHCF.*

A different approach is to overlap the two solving stages, substitution and triangularization. In this algorithm (denoted as *PHCF2*), when a new element of the Cholesky factor is computed, the right-hand side matrix is immediately updated. Thus, two segments of size $(p-1)$ and $2(p-1)$ are simultaneously circulating among the processors and possible idle times are reduced.

We also propose a different approach to increase the granularity of the algorithm while maintaining the initial data distribution. In our algorithm *PHCR* each processor computes at each step q elements of the same row. Due to the data dependency of Hammarling's algorithm, we cannot separate the substitution and triangularization stages as was previously done in algorithm *PHCF*. In this algorithm, $\mathbf{y}$ is actually a matrix of size $n \times q$ and the circulating segment is of size $3q(p-1)$.

4 Cyclic Wavefront Algorithms

Following the analysis by Eisensat *et al.*, we have developed similar pipelined and short-cut algorithms that are variants of those in [5]. The algorithms in [5] were designed for a type of multiprocessor systems where the ratio between the computational speed of the processors and the communication bandwidth of interconnection network was low. Current parallel architectures have experimented an increase of computational speed higher than that of the communication bandwith. On the other hand, many current bus-based multiprocessor systems have reduced the number of processors (4-12) since a high number leads to a dramatic reduction in the performance.

We propose a new sort of parallel algorithms, denoted as cyclic wavefront algorithms (*PHCWF*), where each processor solves simultaneously r columns, and r segments are circulating among the processors (with $r \leq p$, $rk = n$, and k a positive integer number). A row cyclic distribution of S and G is used, and the solution matrix is also stored cyclically by rows. Thus, the Lyapunov equation is solved by using an antidiagonal wavefront of size r. When $r > p$, there are only a maximum of p segments circulating among the processors. The segments will be sent either as a unique message of size $3(p-1)$ or as two segment messages of size $(p-1)$ and $2(p-1)$.

5 Experimental Results

These parallel algorithms have been implemented on an SGI Power Challenge (PCh). This computer reflects a current tendency in the construction of high performance computers. The PCh is a shared memory bus-based multiprocessor (the main memory has 1 GByte) with 12 superscalar R10000 processors at 200 MHz, and 64kBytes and 2MBytes per processor of primary and secondary cache memory, respectively.

The parallel algorithms have been implemented using C and the PVM message-passing library. Communications are tuned and implemented through the main memory. The parallel algorithms were compared with a serial block version of Hammarling's algorithm. We used double-precision arithmetic in our experiments and all the algorithms were compiled with the appropriate optimization flags.

The cyclic algorithms *PHCF*, *PHCF2* and *PHCR* report a low performance as in the case of cyclic triangular linear systems solvers [5, 13]. As we found out in the theoretical analysis of our algorithms, idle times are not reduced when the granularity is increased.

The only way to reduce idle times is increasing the number of segments circulating among the processors while taking advantage simultaneously of data locality in the cache memory. The latter can be achieved by using cyclic wavefront algorithms.

Fig. 4 shows the efficiencies obtained for algorithm *PHCWF* on the PCh using 4 and 8 processors. In this implementation, for each column of the solution that is computed, two segments circulate among the processors.

In this figure, r is the number of columns solved simultaneously (there are $2r$ segments circulating); when $r = 1$ the behavior of *PHCWF* is equal to that of algorithm *PHCF2* (performance in this case are lower than the performance obtained for *PHCWF* with $r = 2$, and therefore these results are not reported). We can observe a low performance for large problems, $r = 2$ and 4 processors. This low performance arises for larger problems and $r = 1$. We are performing further experiments to find out the reasons for this behavior.

Notice that a high performance is obtained only for large-scale problems (the reasons are the large size of the cache memories and the high cost of the communication start-up). It is also possible to observe that the efficiencies obtained are very similar when r is higher than a given value.

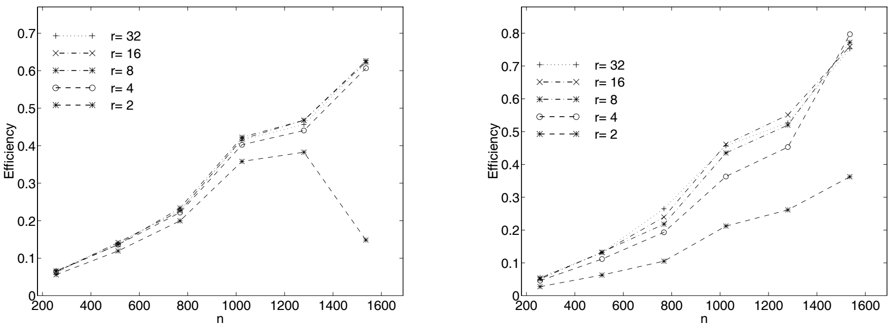

Fig. 4. Efficiency obtained of the algorithm *PHCWF* on the PCh using 4 (left) and 8 (right) processors, for different problem sizes and values of r.

Fig. 5 shows that, in some cases, a higher performance is obtained when the number of processors is increased. Hence, cyclic wavefront algorithms have better scalability than wavefront algorithms designed to solve Lyapunov equations [4], in which the efficiency decreases as the number of processors is increased.

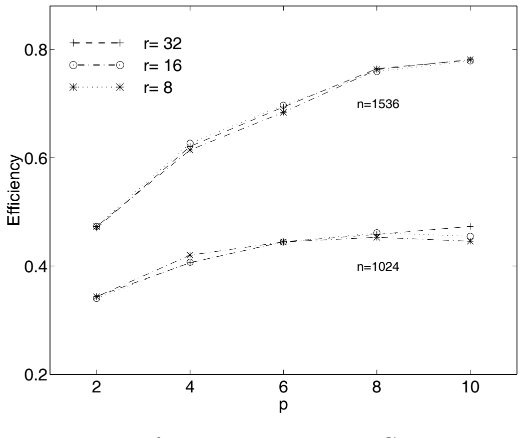

Fig. 5. Efficiency obtained of the algorithm *PHCWF* on the PCh for $n = 1024$ and $n = 1536$, using different number of processors and values of r.

6 Conclusions

New parallel cyclic wavefront algorithms have been presented for the solution of large and dense discrete-time Lyapunov equations using Hammarling's method on message passing multiprocessors. These algorithms can be easily adapted to the continuous-time version of the Lyapunov equation.

The algorithms combine two techniques previously used to solve triangular linear systems and introduce new ideas to solve problems arising in the algorithms.

Cyclic wavefront algorithms show a good performance when the problem order and the number of processors is increased. Efficiencies near to 80% have been obtained for 10 processors and $n = 1536$. This behavior has been tested on a bus-based multiprocessor, the SGI Power Challenge, and excellent scalability has been reported for these algorithms.

References

1. Bai, Z., Demmel, J.: On a block implementation of Hessenberg multishift QR iteration. Int. J. of High Speed Computing **1** (1989) 97–112
2. Bartels, R., Stewart, G.: Algorithm 432. Solution of the matrix equation AX+XB=C [F4]. Comm. ACM **15** (1972) 820–826
3. Claver, J., Hernández, V., Quintana, E.: Solving discrete-time Lyapunov equations for the Cholesky factor on a shared memory multiprocessor. Parallel Processing Letters, **6** No 3 (1996) 365–376
4. Claver, J., Hernández, V.: Parallel wavefront algorithms solving Lyapunov equations for the Cholesky factor on message passing multiprocessors. The Journal of Supercomputing **13** No. 2 (1999) 171–189
5. Eisenstat, S., Heath, M., Henkel, C., Romine, C.: Modified cyclic algorithms for solving triangular systems on distributed-memory multiprocessors. SIAM J. Sci. Statist. Comput. **18** No. 3 (1988) 598–600
6. Golub, G., Nash, S., Van Loan, C.: A Hessenberg-Schur method for the problem AX+XB=C. IEEE Trans. A. C. **24** (1979) 909–913
7. Hammarling, S.: Numerical solution of the discrete-time, convergent, non negative definite Lyapunov equation. System & Control Letters **17** (1991) 137–139
8. Heath, M., Romine, C.: Parallel solution of triangular systems on distributed-memory multiprocessors. SIAM J. on Sci. Statist. Comput. **9** No. 3 (1988) 558–588
9. Henry, G., Watkins, D., Dongarra, J.: A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures. Lapack Working Note #121 (1997)
10. Hodel, A., Polla, K.: Parallel solution of large Lyapunov equations. SIAM J. on Matrix Anal. & Appl. **18** (1992) 1189–1203
11. Kågström, B., Poromaa, P.: Distributed block algorithms for the triangular Sylvester equation with condition estimator. Hypercube and Distributed Computers (1989) 233–248.
12. Laub, A., Heat, M., Paige, G., Ward, R.: Computation of system Balancing transformations and other applications of simultaneous diagonalization algorithms. IEEE Trans. A. C. **32** (1987) 115–122
13. Li, G., Coleman, T.: A new method for solving triangular systems on distributed memory message-passing multiprocessors. SIAM J. Scientific & Statistical Computing **10** (1989) 382–396
14. Moore, B.: Principal component analysis in linear systems: Controlability, observability, and model reduction. IEEE Trans. A. C. **26** (1981) 100–105
15. O'Leary, D., Stewart, G.: Data-flow algorithms for parallel matrix computations. Comm. ACM **28** (1986) 840–853